DOCUMENT RESUME

ED 482 075 IR 058 761

AUTHOR	Breiteneder, Christian; Platzer, Hubert; Hitz, Martin
TITLE	A Re-Usable Software Framework for Authoring and Managing Web Exhibitions.
PUB DATE	2001-00-00
NOTE	15p.; In: Museums and the Web 2001: Selected Papers from an International Conference (5th, Seattle, Washington, March 15-17, 2001); see IR 058 756.
AVAILABLE FROM	Archives & Museum Informatics, 2008 Murray Ave., Suite D, Pittsburgh, PA 15217; e-mail: info@archimuse.com; Web site: http://www.archimuse.com/. For full text: http://www.archimuse.com/mw2001/.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	Authoring Aids (Programming); *Computer Software; *Computer Software Development; Databases; *Exhibits; Foreign Countries; Hypermedia; *Museums; World Wide Web
IDENTIFIERS	Austria (Vienna); *Virtual Museums

ABSTRACT

In the course of a Web development project for the Museum fur Volkerkunde in Vienna, ViEx, a reuse framework supporting authoring and managing hypermedia exhibitions, has been developed. The framework consists of three major components: a relational content database, a corresponding browser based editing interface, and a Web page generator which creates the final Web exhibition. Strict separation of content, structure and layout information promises ease of maintenance, especially in the context of multiple versions of the same exhibition to cope with different presentation languages and client platform dependencies. This paper discusses goals and motivation for the development of the framework; presents the framework itself; puts the framework development into the perspective of the project history; and discusses some advantages and drawbacks of the approach taken. In a final section the authors' current work related to ViEx is briefly described. Includes three tables and six figures. (Author/AEF)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 482 075

A&MI
Archives & Museum Informatics
2008 Murray Ave
Suite D
Pittsburgh, PA
15217 USA
info@archimuse.com
www.archimuse.com

Updated: 02/27/2001 11:35:24

PAPERS
Museums and the Web 2001

A Re-Usable Software Framework For Authoring And Managing Web Exhibitions

Christian Breiteneder, Hubert Platzer , Vienna University of Technology, Martin Hitz, University of Klagenfurt, Austria

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

D. Bearman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

1

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Abstract

In the course of a Web development project for the *Museum für Völkerkunde* in Vienna, ViEx, a reuse framework supporting authoring and managing hypermedia exhibitions, has been developed. The framework consists of three major components: a relational content database, a corresponding browser based editing interface, and a Web page generator which creates the final Web exhibition. Strict separation of content, structure and layout information promises ease of maintenance, especially in the context of multiple versions of the same exhibition to cope with different presentation languages and client platform dependencies.

Keywords: software re-use, exhibit database, authoring tool, virtual exhibition

Introduction

Authoring and managing a hypermedia exhibition is a tedious and time-consuming task requiring a set of skills not readily available in a museum context. In order to liberate curators and exhibition designers from the computer-centric tasks as much as possible and to reduce the resources spent for every new virtual exhibition, we developed ViEx ("virtual exhibition"), a software framework intended to introduce systematic reuse into the domain of Web based exhibitions.

The framework is based on building blocks that cover all structural parts of a Web document, e.g. title, various paragraph types, special character types, as for example, proper names and foreign words and media types like videos, panoramas or images. Some of these types are complex objects themselves, since they consist of a set of more atomic entities. Layout information is available in form of templates that may be nested to define a more complex page.

All building blocks and their instances are stored in a relational database, and pages may be rendered dynamically according to their content and layout definition. In addition, the framework comprises modules allowing for the evaluation of user satisfaction and behaviour, like, for example, guest book and a comprehensive statistics module and a special navigation module allowing for the pictorial representation and arrangement of users' favorite pages.

IR058761

BEST COPY AVAILABLE

ViEx has been developed and tested within the project of a virtual exhibition on Bhutan (http://www.bhutan.at). In principle it has been designed to be platform independent. The current server is running under Solaris and utilizes PHP and Oracle8i. The reusability of the framework is currently being tested in the context of a site for Archaeology@Austria.

The remainder of this paper is organized as follows: In the following section, goals and motivation for the development of the framework are discussed. In Section 3, the framework itself is presented. Section 4 puts the framework development into the perspective of the project history and discusses some advantages and drawbacks of the approach taken. Finally, in Section 5, our current work related to ViEx is briefly described.

Goals and Motivation

Creating an exhibition web site for a museum is a major software development undertaking which may easily go beyond the resources and capabilities of a small or medium-sized museum. For example, the virtual exhibition project, Bhutan -- Fortress of the Gods, (http://www.bhutan.at, Breiteneder et al., 2001) which gave rise to the development framework presented in this paper involved a set of project specific roles. From the category of staff and the number of physical people assigned to each role (stretched to varying degrees), we can easily conclude that - at least as of today - such a project falls outside the main competence and self-image of a museum.

Table 1: Staffing overview for http://www.bhutan.at

Role in Project	Category	#
Project management	Development team	1
Text authoring	Museum	2
Content advisor	Museum	1
Quality assurance	Museum	1
Translation	Translators	4
Screen design	Designer	1
Page editing	Museum	1
Picture editing	Designer + Museum	2
Video editing	Development team	1
3D modeling	Development team	1
Panorama modeling	Development team	1
Software design	Development team	3
Software implementation	Development team	5
Test (functionality, usability)	Development team	5
System administration (development site)	Development team	1

Outsourcing as the obvious solution to the problem of lack of human resources complicates communication during the development phase and impedes ongoing content maintenance after the development phase, as most maintenance requests will have to be directed to the development team which - as a (not unlikely) worst case scenario - may not even exist as such anymore.

In this situation, employing a reuse framework such as ViEx (presented in the next section) may result in the following advantages:

- It enables the museum-based project team together with the art designer to elaborate a complete specification of the exhibition site within the framework provided. This avoids the need for continuous feedback from the development team, thus eliminating significant portions of the communication overhead and focusing the development team on its main task, namely implementing the site according to the specifications provided.
- It shifts development work to a higher level of abstraction: instead of "coding" Web pages in HTML, the main tasks consist of editing and structuring content liberated from layout questions.
- It supports content maintenance in the post-development phase, i.e., it enables museum staff to consistently modify the exhibition without the need to resort to the development team.
- It simplifies the software development portion of the project due to "pre-fabricated" built-in features which can be used as-is and need not be implemented from scratch (concept reuse). For instance, the interplay between images, pop-up legends, and zoom-windows for higher resolution versions of an image is generated by the framework.
- It allows systematic reuse of content building blocks in different contexts within the same virtual exhibition (data reuse). Moreover, export interfaces can easily be established in order to reuse content material outside the project (e.g., consider XML exports).
- By all the above, it saves development time and costs.

The main benefit, however, will be the fact that a framework enforces well-structured and consistent building blocks for the Web presentation. Considering again size and complexity of the project team as sketched in Table 1, any site composed of individual, more or less "hand made Web pages is nearly guaranteed to end up in complete chaos with the typical symptoms of inconsistent hyperlinks, aberrations of page layout etc. The corresponding maintenance overhead - in this case arising already during the development phase - will consume essential project resources. From the authors' experience, such effects are likely to affect even mid-sized projects of about 20 to 50 Web pages.

The advantages discussed above will be especially beneficial in the case of *temporary* virtual exhibitions where high start-up costs may challenge the feasibility of the entire project.

In the specific case of a database-supported framework like ViEx, the existence of an object database containing descriptions and media files for all exhibits needed in the exhibition under construction would constitute an important additional asset, as a significant portion of editorial work could be settled by a straightforward export-import script pre-loading the framework's project database.

ViEx -- A Framework for Virtual Exhibitions

The ViEx framework consists of three major components: a content management system, its underlying content database, and a set of layout templates. In addition, a parser was also implemented to support the transition between the older file system based framework ViEx^{FS} (cf. Section 4) to the current framework operating on a relational database, by transforming Web page files into sets of corresponding database

entries. This auxiliary component may also be considered a "feature" of ViEx because it can be seen as an alternative input channel into the content database.

This section describes the ViEx framework in some detail. We begin with a brief summary of the features of a ViEx-based exhibition and present the main components of ViEx in subsequent subsections.

Supported Features

Any software framework designed for reusability will be more effective, the better its application domain is understood and the more specific the framework fits its application domain (cf. standard literature on software reuse, e.g. Biggerstaff & Perlis, 1989). Thus, when employing such a framework, the trade-off between generic applicability and reuse potential must be carefully considered. To this end, it is important to briefly summarize the features of a virtual exhibition realized on the basis of ViEx. For a more in-depth treatment see Bhutan - A Virtual Exhibition http://www.bhutan.at.(Breiteneder et al. (2001).

The content of a ViEx exhibition is organized in a layout-independent manner. Building blocks may be *pages* (content aggregates consisting of other building blocks), *pictures* with descriptive information, *video*, *audio*, *text* (individual text and standard text building blocks), *glossary entries* (currently two categories: foreign words and proper names), *script building blocks* (e.g., fader texts), and *panoramas* (possibly linked to descriptive information). Table 2 summarizes major attributes assigned to these building blocks. For all text-based information, the respective language is another attribute stored in the database (not shown in Table 2).

Table 2: Content building blocks and their major attributes

Content Building Block	Attributes
Page	type of page (chapter, subchapter...), template, window title, document title, color code, ...
Picture	Image URL, zoom image URL, title, legend, display size
Video, Audio, Panorama	URL, title, legend, plug-in, size, quality...
Text	type (i.e., style class like header, paragraph...), content
Glossary entry	text, category, short description ("tool tip"), long description (glossary page)
Script building block	text

As navigation mechanisms, ViEx supports manually defined *ad hoc hyperlinks*, systematic *hierarchic links* (path to the current page, children

emanating from the current page etc.) for navigating through the exhibition according to the underlying book metaphor, *local links* to navigate within a page, a *site map* containing thumbnails of the upper levels of the hierarchy, a *hierarchy browser* (available in three flavors: pull-down-menus, table of contents, tree controls), a *search facility*, an *exhibit index*, *pop-up windows* displaying descriptive information linked to pictures, and *zoom windows* displaying pictures at a higher resolution. In addition, there are two different tour concepts available: a *slide show* consisting of a predefined sequence of pictures with legends which are continuously presented (fading over each other), and a *guided tour* taking a sequential path through the exhibition, briefly explaining its different parts. Last but not least, registered users get access to a feature called *lookmarks* enabling them to create their own *catalog* of the exhibition by distributing thumbnails of pages of interest in a three dimensional space and grouping them according to subjective organizing principles. (The three dimensional arrangement and the visual cues based on thumbnails are deemed to provide a much better recognition / retrieval rate compared to ordinary bookmarking.)

The book metaphor mentioned above serves as the major structuring mechanism in ViEx. Its strict hierarchic structure is deemed to reduce the likelihood for a visitor to the virtual exhibition to get "lost in hyperspace", especially since visitors are assumed to be able to relatively quickly construct a mental model of the Web presentation due to their acquaintance with exhibition catalogues and the like.

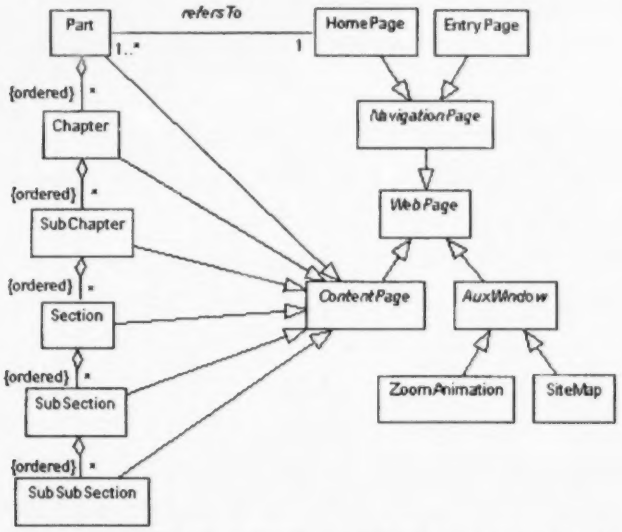


Fig.1: Relationships between ViEx web pages (in UML)

The six (at most -- note the cardinalities indicated by * in the UML diagram in Fig.1) hierarchy levels represent the main information-bearing content pages. Besides them, a few other, secondary variants of the abstract concept "web page" exist for rather technical purposes.

ViEx supports multiple coexisting versions of the same exhibition. Specifically, three versioning dimensions can be distinguished:

- Static vs. dynamic pages: During the presentation design phase, pages generated on demand are usually preferred to static

BEST COPY AVAILABLE

pages, because changes in the "look & feel" of the site are immediately made visible on all available pages for persons working on the design prototype. When the design has stabilized, one may want to switch to static pages either for performance reasons or in order to prepare an off-line version of the exhibition on CD-ROM. Dynamic ViEx pages are coded in PHP.

- Different technological levels: Taking into account the diversity of client platforms now in use and their mutual incompatibilities, it is necessary to customize a Web presentation to several levels of browser technology. As a minimum requirement, a "high tech" version with contemporary ("cool") features and a 'low tech" version suited to old or low-end browsers should be provided.
- Language of presentation.

Content Database

The content database is implemented using a relational database management system and contains both the *content* and *structure* of the virtual exhibition, but excludes low-level layout information. This separation represents a standard design goal for complex Web sites, because it guarantees consistent presentation of the content throughout the whole Web site. The database oriented approach of ViEx pushes this design goal to an extreme: As no low-level formatting information can be entered into the content database, the separation of content and layout follows as a consequence.

The data model underlying the content database is given as a UML diagram in Fig. 2 . As can be seen from Fig. 2 the core concept of the content database is the abstract entity *building block* which represents a unit of information to be presented and manipulated as a whole. The specialization hierarchy emanating from class*BuildingBlock* distinguishes between atomic entities and complex entities represented by containers consisting of other building blocks. Atomic entities may be:

- homogeneous portions of text (with no intermingled markup) in various paragraph types and optional special character distinction (e.g. for proper names and foreign words),
- pictures consisting of a small image, a caption, a more detailed description and a reference to a larger copy
- other media like videos and panoramas treated in a similar way (not shown in the figure), and
- various kinds of hyperlinks.

A content page is associated with an externally stored layout template divided into so-called "slots" (class *Content* in Fig. 2) which are to be filled by building blocks. Slots may in turn be subdivided into "subslots", where the layout of each subslot is again defined by a layout template. The hierarchy specified in Fig.1 is resolved as a recursive parent-child relationship between *content page* entities. For each content page, some layout information in the form of attribute-value pairs (such as color schemes or main titles) can be specified and is propagated down the page hierarchy by the page generator (hence the class name *InheritanceData*). Thus, every page in the hierarchy takes this information from higher level pages, but it can also be assigned corresponding data of its own which overrides the "inherited" default.

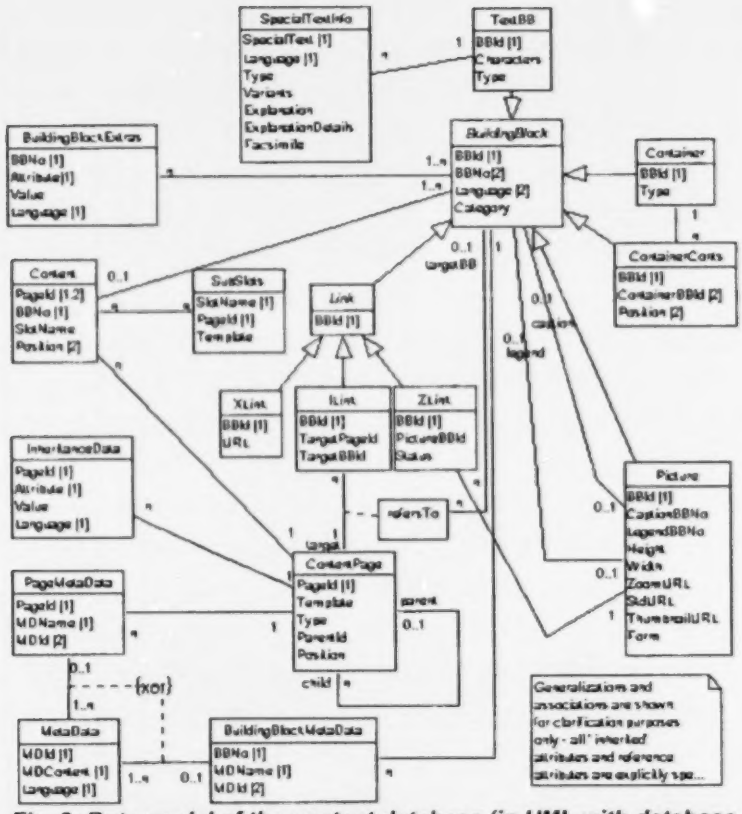


Fig. 2: Data model of the content database (in UML with database key designators in brackets)

When the exhibition is generated from the content database, the resulting page format is determined by

- the position within the page hierarchy at which the page resides which affects the inheritance mechanism explained before and the generation of structural navigation links,
- the layout template assigned to the page,
- templates possibly assigned to the slots of the template,
- a cascading style sheet (which is dynamically selected depending on the client's platform),
- the formatting strategy encoded into the generator for each type of building block,
- as well as additional layout information assigned to selected building blocks (class *BuildingBlockExtras*).

Front end (input and editing interface) and back end (pagegenerator) of the content database together form the content management system described at the end of this section.

Layout Templates

Layout templates reside in the file system (outside the content database) and define the principal layout of each type of page. Changing a layout template modifies the look & feel of the exhibition without modifying its

BEST COPY AVAILABLE

content and logical structure.

Templates are "stub pages" with fixed HTML, includes etc. combined with special markups designating slots to be filled from the content database. The syntax of these markups is

<!-- #BeginEditable "SlotId" -->

and is taken from MacroMedia Dreamweaver in order to allow employing Dreamweaver as a comfortable template editor. The generator replaces this markup with the building blocks selected from the database via slot identification *SlotId* and the identification of the page under construction.

Besides the content oriented slots mentioned, there are also so-called "technical slots" with predefined semantics which are used to designate the position of standard page elements like headings and navigation blocks within the page.

For instance, a typical template taken from http://www.bhutan.at contains the following slots:

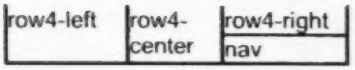
a) Technical slots

- *Page title*: The browser's window title. Content explicitly given in the content database.
- *Page meta-information*: HTML meta information (invisible). Content explicitly given.
- *Script*: Includes client side scripting files (e.g., JavaScript) with functions taking care of CSS selection according to the client platform, window management, and pop-up handling. Content generated.
- *Document title*: Text or image. Content explicitly given.
- *Color scheme*: Color code or image. Content explicitly given or inherited from ancestor pages.
- *Path*: Navigation element consisting of titles of all ancestor pages. Content implicitly given (taken from ancestor pages).
- *Navigation* ("nav", cf. the table below): One of two possible types of hierarchical navigation menu depending on the type of page. Content implicitly given.
- *Pop ups*: Place holder for pop-ups used as image legends. Must be last element in the template. Content generated.

b) True content slots: The standard layout for "Bhutan -- Fortress of the Gods" is based on the following HTML table (with slot names):

Table 3

row1-left	row1-center	row1-right
row1		
row2-left	row2-center	row2-right
row2		
row3-left	row3-center	row3-right
row3		



All these slots may be assigned simple text building blocks, containers holding nested building blocks, or subslots with corresponding subtemplates. This latter feature enhances the range of application of a template: Instead of designing completely new templates, many variations can be incorporated into one and the same template by means of the subtemplate concept.

For technical reasons, ViEx only supports one level of subtemplates; i.e., subtemplates nested within subtemplates are not possible. However, for the virtual exhibition "Bhutan --Fortress of the Gods" this restriction did not impose any undue design limitations.

It should be noted that all content which depends on the page type only, i.e., which is identical for all pages using the same template, might also be kept in the template itself (and not in the content database). However, such an approach may jeopardize important features of the framework such as creating a presentation in a different language etc.

Content Management System

The content management system (CMS) consists of the content database input and editing interfaces and the page generator. It allows creation, maintenance and deletion of pages. In its current version, it supports ODBC, Oracle's OCI and MySQL databases, thus covering all major production platforms.

The *database input and editing interface* is browser based and provides features to manipulate content, structure and layout of the web presentation under construction by modifying respective database contents. Fig. 3 shows a screenshot of the hierarchy editor based on tree controls and an editing menu. This component is used to navigate and/or modify the hierarchic structure of the exhibition. From here, the main content edit window may be reached. In Fig. 4 , building blocks 1, 2, and 3 of Container 11 (itself an aggregate building block of type "baui") of page "Infrastruktur – Baudenkmäler" of a German web page are being edited. The structure of the edit window changes dynamically depending on the type of content edited.

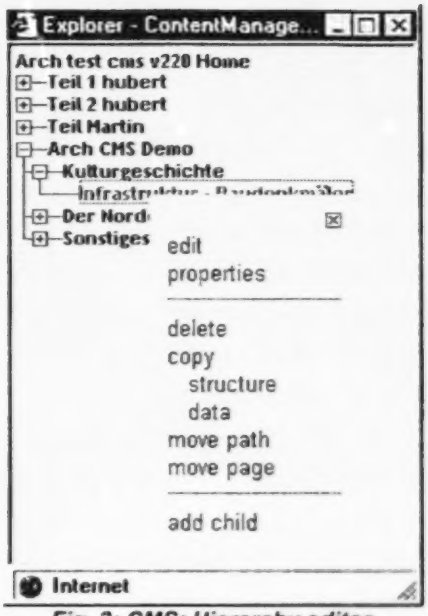


Fig. 3: CMS: Hierarchy editor

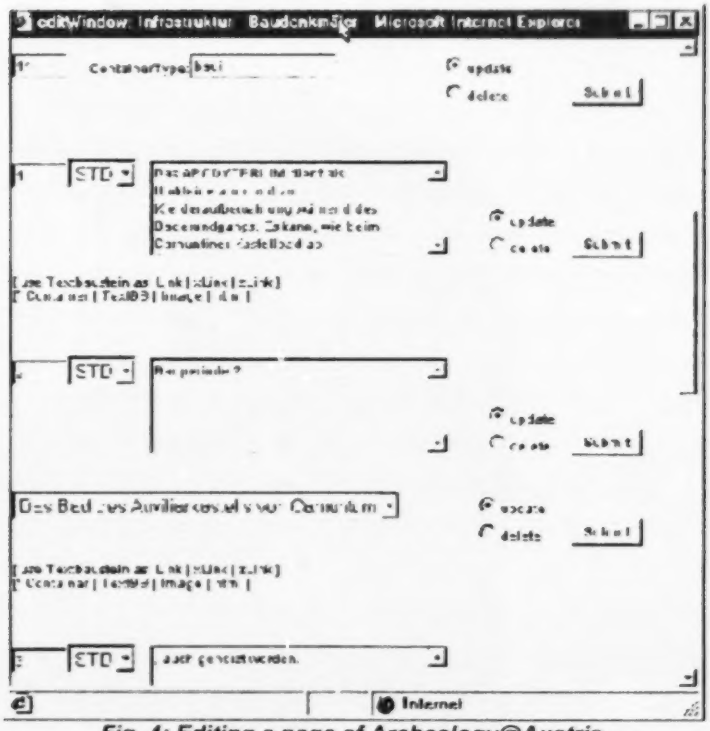


Fig. 4: Editing a page of Archeology@Austria

Fig. 4 illustrates not only the look & feel of the user interface for content entry but also the main disadvantage of the required pseudo-atomicity of the content database: As there is no formatting allowed *within* text building blocks, the granularity of text building blocks is somewhat

BEST COPY AVAILABLE

smaller than mere re-use considerations would require. For instance, as text portion "*Bauperiode 2*" is used as a hyperlink to page titled "*Das Bad des Auxiliarkastells von Carnuntum*", it must be entered as a separate building block (no. 2), although from a semantic perspective, building blocks 1, 2, and 3 together form a logical unit (i.e., a paragraph).

As mentioned before, it is possible to associate some layout information will different types of building blocks which will be taken into consideration by the page generator. The corresponding CMS module (style editor) is shown in Fig. 5 .

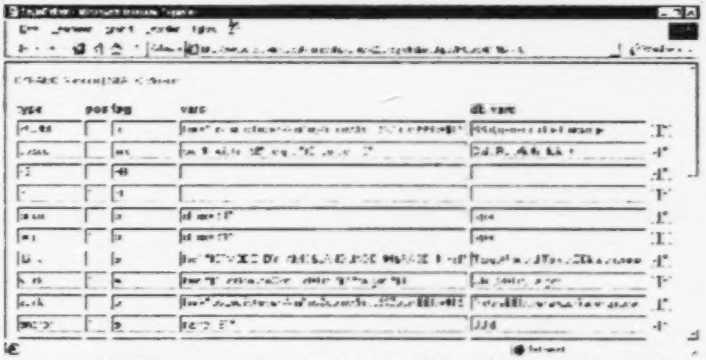


Fig. 5: Editing styles associated with building block types

Columns "tag" and "vars" define the HTML element to be generated for building blocks of type "type". "vars" contains variable references ("$1", "$2", etc.) which are to be substituted by corresponding database attributes listed in column "db-vars". These attributes are found elsewhere in the database. (Column "pos" is relevant for nested blocks only.)

Last but not least, a preview window allows the user to immediately see the outcome of the page generator run (Fig. 6). In such a window, special icons offer links back to specific edit windows (edit picture, edit container, edit text building block, edit styles, edit template slot. Cf. the bracketed icons in Fig. 6 -- clicking on the one with the tool tip displayed would open the picture editing window to modify the presentation parameters of the image of Guru Rinpoche).

BEST COPY AVAILABLE

Fig. 6: Preview of generator run

The *page generator* represents the back end of the content management system, producing dynamic Web pages (a static version can be generated on demand in a post-processing step) according to the formatting rules given earlier. It is responsible for a homogeneous treatment of the different kinds of building blocks (e.g., generating the code necessary for the interplay between images, captions, corresponding pop-up legends and zoomwindows), and consistent navigation scheme, both within a page and within the web site (for deep hierarchies, managing the complex navigation menus is in itself a nontrivial task which is hard to accomplish manually, especially when the structure of the hierarchy is changed).

The generator is also capable of creating different versions of an exhibition as discussed in Section 3. For example, "Bhutan -- Fortress of the Gods" (http://www.bhutan.at) exists in German and English, both in a "high tech" and "low tech" variant.

Project History and Reflection

The project which gave rise to the development of the framework presented in this paper begun as a research project to explore different approaches to virtual exhibitions, funded by the Austrian Federal Ministryof Education, Science and Culture (Breiteneder et al., 2000). However, as a by product, a production release of a virtual variant of a well-known and internationally successful exhibition of the Austrian *Museum für Völkerkunde* on Bhutan had to be developed. In this situation, the project partners opted for a dual approach: A conventional development track based on manually created dynamic Web pages was chosen in parallel to the development of the more advanced database based infrastructure presented in this paper (ViEx^{DB} or ViEx, for short). The first track was needed to guarantee a working product, while the second track was aimed towards a reuse environment for future virtual exhibitions.

During the development of the Web site along the conventional track, strong emphasis was laid on a well structured project and software design suited for a mid-sized project. Thus, after a short exploration phase, a set of tools (PHP3 as scripting language, homesite and BBedit

BEST COPY AVAILABLE

as web editors, m4 as macro processor to introduce parametric text substitution etc.) to be employed alongside with corresponding strict coding and naming conventions were established, and the file system based version ViEx^{FS} of the framework was born. When its design had matured so far that a successful completion of the development project could be foreseen, all structural decisions made were frozen and the database oriented approach was started in parallel. From the modeling decision made thus far, the database scheme was deduced and the page generator was developed. Meanwhile, conventional development of the web site was progressing within the ViEx^{FS} framework, so that a content transfer tool was needed to initialize the database content from the web pages in ViEx^{FS}. The development and employment of a respective page analysis tool helped to verify and optimize the database model and also pointed out some inconsistencies within ViEx^{FS} which were corrected on the fly. Then, the editing front end was implemented as a working prototype, and finally the whole Web site was loaded into the database version of ViEx and manually polished. From this dual development experience, we are now able to contrast the two approaches taken.

In the file-system based version of the development framework, each building block was realized as a file in directory tree with strict localizing and naming conventions. The unit of work was defined as editing such a source file which usually took place on a client followed by some kind of data transfer (e.g., ftp) to the web server. With a Web-based editing interface, some edits could also be done directly on the Web server. Thus, a maximum of content creation flexibility was achieved, and all file system tools available on the server (under Unix) could be used (e.g., access right configuration, global search and replace etc.). In particular, pages systematically generated via PHP3 and m4 could easily be combined with some hand crafted individual pages where necessary. Also, authoring and configuration management activities could evolve with the complexity of the web site under construction.

The disadvantage was that many interfaces had to be mastered by the editing staff and that there was no real control whether all conventions established by the project team were observed (as the main portion of the content editing work was done by a single person with sufficient self-discipline, this aspect did not create a lot of problems in our project). Moreover, data transfer from the client workstation to the web server was not secured.

In the database version of the framework, content was moved from individual files into tables of a relational database system. The corresponding conceptual model guaranteed clear separation of content, structure and layout. There is only one editing interface (cf. the previous section). Modest use of relational features allows the employment of a wide range of DBMS from mySQL to Oracle 8. As the editing interface is somewhat more formal than for ViEx^{FS}, there were fewer accidental changes of content. On the minus side, there is the higher complexity of the editing task (due to the atomic nature of data), even though content editing performance seems to constitute a critical success factor for such a project. We also had to realize that late modifications of the underlying conceptual model cost a relatively great deal of work (an experience which seems counter-intuitive to classical arguments in favor of database technology). Thus, the database-oriented approach seems to be especially promising if:

- the structure of the planned exhibition is compatible to the framework, i.e., it can be mapped onto the conceptual model of the framework,
- the exhibition project exceeds a certain complexity which allows amortization of setup costs (installing and getting acquainted with the tools, etc.),
- all tools are already well-defined and ready for duty at the beginning of the project,
- an exhibit database exists which may be linked to the content database presented here.

Conclusions and Ongoing Work

The primary project goal to develop http://www.bhutan.at to a fully functioning Web site meeting all requirements of a production system has put restrictions on the development of the prototype of a reuse framework discussed in this paper. We will definitely need at least resources equal to those spent so far on its completion to be able to use it in full practice and in a wide variety of contexts. Since reuse of software can only be proven by reuse, we have meanwhile tested the framework in a feasibility study in the field of archaeology. Because the participating archaeologists had different requirements from the ones to be met for http://www.bhutan.at, minor adjustments were necessary to increase the functionality available.

Currently, we are extending ViEx to be prepared for a much larger project covering all Austrian archaeological museums and sites in one Web portal. Since the total number of Web pages of this site will be a magnitude higher than for Bhutan, we first have to evaluate and improve the performance of ViEx for sites with more that 1000 pages. The most important additional extensions of ViEx will cover:

- support for XML,
- support for metadata (RDF and Topic Maps) in order to move an important step towards the *semantic web*,
- support for multiple concurrent users and
- the enhancement of editing features, as for example the splitting of building blocks.

Moreover, we want to compare available content management systems and to include some of the convincing additional features in our system.

References

Breiteneder, C., Hitz, M., Hon, M., & Platzer, H. (2000). Untersuchung innovativer Hypermedia-Methoden zur Gestaltung virtueller Ausstellungen am Beispiel "Bhutan -- Festung der Götter. http://www.bhutan.at. Project report, Vienna University of Technology/ University of Klagenfurt / University of Vienna, October 2000.

Breiteneder, C., Hitz, M., Platzer, H., & Stockinger, J. (2001). Bhutan -- A Virtual Exhibition. In D. Bearman & J. Trant (Eds.) *Museums and the Web 2001 Selected Papers from an International Conference*. Pittsburgh, Archives & Museum Informatics, 2001.

Biggerstaff, T. J. & Perlis, A. J. (1989). Software Reusability (Vol. I: Concepts and Models, Vol. II: Applications and Experience). ACM

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

X	This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.
	This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").